Heterogeneous Style Composition Analysis

Cristina Gacek

March 10, 1998

Outline

- Introduction
  - Motivation
  - Problem Statement and Approach

- Current Results
  - Styles Investigated
  - Conceptual Features
  - Architectural Mismatches

- Example
- Conclusion
Motivation

- Component-based software engineering
- Architectural mismatch problems
- Architectural styles
  - Constrains design elements and their relationships
  - Characterized by various features
    - Features may be used to describe ANY system
  - Potential mismatches
    - Homogeneous styles
    - Heterogeneous styles

Problem & Approach

"What is a formal model for detecting potential architectural mismatches during the composition of different software systems?"

- Styles descriptions
- Conceptual features relevant to composition
- Dimensions of conceptual features
- Model of styles
- Architectural mismatches
- Model of systems composition
- Tool
Styles Investigated

- Pipe-and-Filter
- Main-Subroutine
- Distributed Processes
- Layered
- Object-Oriented
- Blackboard
- Event-Based
- Rule-Based
- Logic Programming
- Transactional Database-Centric
- Real-Time
- Closed-Loop Feedback Control
- Internet Distributed Entities

Architectural Styles Characteristics

- Pure architectural styles:
  - Easily understood and manipulated
  - Usually not enough for large software systems
- Shared constraints among styles
  - Conceptual features
- Set of constraints can be used for specification
- Use of features for mismatch detection
Conceptual Features

- Dynamism
- Supported Data Transfers
- Triggering Capability
- Concurrency
- Distribution
- Layering
- Encapsulation
- Termination

Conceptual Features (cont.)

- Preemption
- Predictable Response Times
- Component Priorities
- Backtracking
- Reentrance
- Reconfiguration
- Central Control Unit
Other Characteristics

- AI related: (need further study)
  - Forward vs. backward chaining
  - Rules preconditions matching vs. unification of head terms

- Database related: (will not be addressed)
  - Single vs. multiple users
  - Roll-back
  - Security and integrity constraints
  - Semantic heterogeneity
  - Data items granularity
  - Database distributed or not
  - Different structural organizations

March 10\textsuperscript{th}, 1998
Architectural Mismatches Classification

- Byproduct of compositional analysis
- Categorized by:
  - Bridging connector
  - Conceptual feature choice
- Partial classification
  - Some others already determined, but not yet classified

Heterogeneous Style Composition Analysis

<table>
<thead>
<tr>
<th>Dynamism</th>
<th>Call</th>
<th>Spokes</th>
<th>Bridging connector</th>
<th>Shared data</th>
<th>Triggered &amp; shared</th>
<th>Trigger shared</th>
<th>Shared &amp; triggered</th>
<th>Shared &amp; triggered</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6</td>
<td>10</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data transfers</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triggered</td>
<td></td>
<td>5,7,9</td>
<td>5,7,9</td>
<td>5,7,9</td>
<td>5,7,9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concurrency</td>
<td>17</td>
<td>13,36</td>
<td>2,3,16</td>
<td>1</td>
<td>17</td>
<td>13,36</td>
<td>15,19</td>
<td></td>
</tr>
<tr>
<td>Distribution</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layering</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Encapsulation</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>20</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Termination</td>
<td>18</td>
<td></td>
<td></td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preemption</td>
<td>37</td>
<td>38</td>
<td>37</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predictable Response Times</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component Priorities</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td>19,29</td>
</tr>
<tr>
<td>Backtracking</td>
<td>30</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>30</td>
<td>31</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>Reentrance</td>
<td>26</td>
<td>27</td>
<td>25</td>
<td>24</td>
<td>26</td>
<td>27</td>
<td>25</td>
<td>19</td>
</tr>
<tr>
<td>Reconfiguration</td>
<td>35,39</td>
<td>35</td>
<td>35</td>
<td>35,39</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td>19</td>
</tr>
<tr>
<td>Central control unit</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td>19</td>
</tr>
</tbody>
</table>
Heterogeneous Style Composition Analysis

Current Mismatches Classification (b)

1. Two concurrent threads share data, with potential synchronization problems.
2. Two threads have data connectors to 2 different control components in a third thread (it is impossible for the third thread to execute in the two components simultaneously).
3. Two control components in the same thread share a blocking data connector, creating a possibility of deadlock.
4. A layering constraint is violated.
5. Different sets of recognized messages are used by two subsystems that permit triggers.
6. A spawn is made into a subsystem which originally forbade them.
7. An unrecognized trigger message is used.
8. A triggered spawn is made into a subsystem which originally forbade spawns.
9. A trigger refers to a subsystem which originally forbade triggering.

March 10° 1998

Heterogeneous Style Composition Analysis

Current Mismatches Classification (c)

10. A data connector is made into a subsystem which originally forbade them.
11. A shared data relationship refers to a subsystem which originally forbade them.
12. A trigger refers to a subsystem which forbids explicit or implicit data connectors, hence the trigger may never occur.
13. A (triggered) spawn is made into a subsystem which is not concurrent.
14. A remote connector is extended into or out of a non-distributed subsystem (i.e. a subsystem originally confined to a single node).
15. A node resource is overused (this is actually checked by summing across the subsystems' usage of that particular resource).
16. Data connectors connecting control components that are not always active may lead into deadlock or loss of data.
17. Erroneous assumption of single-thread.
18. (Triggered) Call to a non-terminating control component.

March 10° 1998
Current Mismatches Classification (d)

19. Erroneous assumption of same underlying platform.
20. (Triggered) Call to a private method.
21. (Triggered) Spawn to a private method.
22. Sharing private data.
23. More than one central control unit exists, all of them assuming they have absolute control on execution sequencing.
24. Sharing data with the incorrect invocation of a component.
25. Incorrect assumption of which invocation of a component is either sending or receiving a data transfer.
26. (Triggered) Call to a non-reentrant component that is already running.
27. (Triggered) Spawn to a non-reentrant component that is already running.
28. Not clear where a component(s) with or without an associated priority fits in an overall separately prioritized structure. The component may be either (triggered) called or (triggered) spawned.

Current Mismatches Classification (e)

29. A prioritized system sharing a machine with some other system. How do priorities compare across systems?
30. (Triggered) Call or spawn from a subsystem that may later backtrack causing possible undesired side effects on the called/spawned end.
31. Data being transferred to or from some component(s) that may later backtrack.
32. Shared data being modified by a component(s) that may later backtrack.
33. (Triggered) Call from a subsystem requiring some predictable response times to some component(s) not originally considered. May have side effects on original predicted response times.
34. (Triggered) Spawn to or from a subsystem requiring some predictable response times. The original predictable response times for S did not account for concurrently executing with these other components.
35. Only part of the resulting system automatically reconfigures upon failure.
Current Mismatches Classification (f)

36. Some component(s) that was (triggered) spawned has different synchronization mechanism than the spawner.
37. (Triggered) Call to a component that should be preemptable and isn't.
38. (Triggered) Spawn to a component that should be preemptable and isn't.
39. (Triggered) Call to a component that performs on the fly garbage collection.

Composition Analysis

1- Describe parts to be composed in terms of features (or pure styles)
   - If dealing with styles, then identify the specific choices for "unconstrained" features
2- Determine which kind(s) of connector(s) will be used to compose them
3- For each feature and the given connector(s) traverse the mismatch table
   - For each mismatch in the relevant column(s) determine which ones are relevant
Outline

- Introduction
  - Motivation
  - Problem Statement and Approach
- Current Results
  - Styles Investigated
  - Conceptual Features
  - Architectural Mismatches

Example

- Conclusion

Style Composition Example

1- Describe parts to be composed in terms of features (or pure styles)

<table>
<thead>
<tr>
<th></th>
<th>Real-Time</th>
<th>Internet Distributed Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamism</td>
<td>threads</td>
<td>threads, downloaded Applets</td>
</tr>
<tr>
<td>Supported data transfers</td>
<td>unconstrained</td>
<td>explicit data connectors</td>
</tr>
<tr>
<td>Triggering capability</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Concurrency</td>
<td>multi-threaded</td>
<td>multi-threaded</td>
</tr>
<tr>
<td>Distribution</td>
<td>unconstrained</td>
<td>multiple nodes</td>
</tr>
<tr>
<td>Layering</td>
<td>unconstrained</td>
<td>yes</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>unconstrained</td>
<td>unconstrained</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Real-Time</th>
<th>Internet Distributed Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Termination</td>
<td>unconstrained</td>
<td>unconstrained</td>
</tr>
<tr>
<td>Preemption</td>
<td>yes</td>
<td>unconstrained</td>
</tr>
<tr>
<td>Response Times</td>
<td>predictable</td>
<td>unbound</td>
</tr>
<tr>
<td>Component Priorities</td>
<td>yes</td>
<td>unconstrained</td>
</tr>
<tr>
<td>Backtracking</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Reentrance</td>
<td>no</td>
<td>yes (server)</td>
</tr>
<tr>
<td>Reconfiguration</td>
<td>unconstrained</td>
<td>on the fly garbage collec-</td>
</tr>
<tr>
<td>Central control unit</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
**Style Composition Example**

- If dealing with styles, then identify the specific choices for "unconstrained" features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Real-Time</th>
<th>Internet Distributed Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamism</td>
<td>yes</td>
<td>threads, downloaded Applets</td>
</tr>
<tr>
<td>Supported data transfers</td>
<td>shared variables and explicit data connectors</td>
<td>explicit data connectors</td>
</tr>
<tr>
<td>Triggering capability</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Concurrency</td>
<td>multi-threaded</td>
<td>multi-threaded</td>
</tr>
<tr>
<td>Distribution</td>
<td>single node</td>
<td>multiple nodes</td>
</tr>
<tr>
<td>Layering</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

**March 10, 1998**

---

**Style Composition Example**

- 2- Determine which kind(s) of connector(s) will be used to compose them
  - Call from the Real-Time to the Internet Distributed System
### Style Composition Example

**3.** For each feature and the given connector(s) traverse the mismatch table

<table>
<thead>
<tr>
<th>Feature</th>
<th>Call</th>
<th>Triggered</th>
<th>Tested State</th>
<th>Shared Memory</th>
<th>Triggered Specifications</th>
<th>Shared Mismatch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denotation</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data transfers</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>Triggered</td>
<td>5,7,9</td>
<td>5,7,9</td>
<td>5,7,9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Concurrency</td>
<td>17</td>
<td>12,36</td>
<td>2,3,16</td>
<td>3</td>
<td>17</td>
<td>13,36</td>
</tr>
<tr>
<td>Distribution</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Layering</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>20</td>
<td>21</td>
<td>19</td>
</tr>
<tr>
<td>Termination</td>
<td>17</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Persistence</td>
<td>37</td>
<td>38</td>
<td>37</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predictable Response Times</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component Priorities</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backwarding</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>30</td>
<td>31</td>
<td>19</td>
</tr>
<tr>
<td>Reversibility</td>
<td>26</td>
<td>27</td>
<td>25</td>
<td>24</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>Persistence</td>
<td>35,19</td>
<td>35</td>
<td>36</td>
<td>35,30</td>
<td>35</td>
<td>30</td>
</tr>
<tr>
<td>Central control unit</td>
<td>22</td>
<td>23</td>
<td>23</td>
<td>22</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

March 1986

---

### Style Composition Example

**✔ 4.** A layering constraint is violated.
- The Internet Distributed system (ID) is layered.

**✔ 14.** A remote connector is extended into or out of a non-distributed subsystem (i.e. a subsystem originally confined to a single node).
- The Real-Time system (RT) was confined to a single node.

**✗ 17.** Erroneous assumption of single-thread.
- Both RT and ID were designed to be multi-threaded.

**✗ 18.** (Triggered) Call to a non-terminating control component.
- ID does terminate.

March 1986
Style Composition Example

✔ 20. (Triggered) Call to a private method.
   - ID has encapsulation.

✘ 23. More than one central control unit exists, all of them assuming they have absolute control on execution sequencing.
   - Only RT has a central control unit.

✘ 26. (Triggered) Call to a non-reentrant component that is already running.
   - ID is reentrant.

✔ 28. Not clear where a component(s) with or without an associated priority fits in an overall separately prioritized structure. The component may be either (triggered) called or (triggered) spawned.
   - RT has components priorities.

✘ 30. (Triggered) Call or spawn from a subsystem that may later backtrack causing possible undesired side effects on the called/spawned end.
   - Neither of them backtrack.

✔ 33. (Triggered) Call from a subsystem requiring some predictable response times to some component(s) not originally considered. May have side effects on original predicted response times.
   - RT requires predictable response times.
Style Composition Example

- 35. Only part of the resulting system automatically reconfigures upon failure.
  - RT automatically reconfigures whereas ID doesn't
- 37. (Triggered) Call to a component that should be preemptable and isn't.
  - ID has no preemption.
- 39. (Triggered) Call to a component that performs on the fly garbage collection.
  - ID does perform on the fly garbage collection.

Outline

- Introduction
  - Motivation
  - Problem Statement and Approach
- Current Results
  - Styles Investigated
  - Conceptual Features
  - Architectural Mismatches
- Example
- Conclusion
Conclusion

- Strong set of conceptual features
  - for describing and comparing architectural styles, as well as systems
- Model of architectural styles
- Model of systems composition
- Conceptual feature space description
- Application to a realistic problem
- Tool that analyzes for architectural mismatches